Manipulando datos gigantes con Pandas

alejandro-gamarra
10/03/2021

Hola amigos, ustedes ya se habrán dado cuenta que cuando trabajamos con datos en Python siempre terminamos utilizando la biblioteca Pandas ¿verdad? Y no es por coincidencia, ya que Pandas es una herramienta de análisis y manipulación de datos de código abierto, rápida, potente, flexible y fácil de usar. Siendo una de las piezas clave de la enorme popularidad de Python entre los científicos de datos. Pero los datos de Pandas se manejan en memoria y, por lo tanto, cuando el tamaño de los datos crece es difícil poder trabajar con ellos. Entonces, ¿sólo podremos trabajar con datos pequeños? ¡Claro que no!, la belleza de esta biblioteca es que también está preparada para enfrentar este problema y ustedes, que están leyendo este blog, aprenderán los secretos para superar este desafío.

¡Divide y vencerás!

Quién ya ha escuchado esta frase sabe de lo que estoy hablando, y trabajar con “Big Data” es exactamente esto, así es mis amigos, el secreto es, dividir un banco de datos gigante en particiones más pequeñas y procesar cada una de éstas independientemente para aprovechar todo el potencial de la biblioteca Pandas sin sobrepasar el límite de la memoria al mismo tiempo. Si no dividieramos, simplemente el total de líneas de la base de datos no cabrían en la memoria disponible.

Dividiendo una base

Cuando trabajamos con una tabla o archivo de gran tamaño, de varios gb. El contenido total del archivo no va a caber en la memoria del ordenador. Por eso lo indicado es particionar. Para ello, existe un atributo en la función **read_sql** ó **read_csv** de Pandas que permite colocar el número máximo de filas que se cargarán en cada partición. Este atributo es el chunksize. De esta manera, estaríamos importando una partición más pequeña de cada vez en un objeto DataFrame de Pandas, y que podrá ser almacenado en memoria y sobre el cual, nosotros podremos realizar cualquier manipulación en los datos. Veamos 3 ejemplos de cómo realizar esta partición:

1.Importando un archivo CSV gigante

Imaginemos que nos compartieron el archivo CSV “rides.csv” que contiene todos los viajes del mes de una conocida empresa de taxis de una aplicación, y que dentro del archivo CSV existe una columna “estrellas” con la calificación del viaje, el pedido es: “Generar un archivo CSV con únicamente los viajes que tuvieron ‘5’ estrellas”, parece una extracción fácil de realizar ¿verdad?, pero ¿cuál es el problema? El archivo tiene 150MM de líneas, veamos cómo el atributo **chunksize** nos ayuda con este problema:

```
# Definimos el tamaño máximo de filas de cada partición (1MM)
import pandas as pd
size = 1000000
# Creamos un objeto iterador 'df_chunk' que contendrá las particiones con 1MM
df_chunk = pd.read_csv('rides.csv', chunksize=size)
# Creamos una variable booleana verdadera 'header' para exportar las cabeceras
header = True

# Ahora vamos a recorrer cada partición, realizar el filtro solicitado, y expo
# El atributo mode='a' (APPEND) sirve para no sobreescribir el archivo Resulta
# Luego de la 1ra iteración 'header' vuelve a ser falsa para no colocar nuevam
for chunk in df_chunk:
    chunk_filter = chunk[chunk['estrellas'] == 5]
    chunk_filter.to_csv('Resultado.csv', header=header, mode='a')
    header = False
```

2. Importando una tabla SQL gigante

Ahora imaginemos que nos han solicitado exportar a TXT el contenido de una tabla SQL “Factura” que contiene toda la facturación del mes de una conocida empresa de celulares, también parece algo sencillo ¿verdad?, pero nuevamente ¿cuál es el problema? Esta tabla tiene 100MM de líneas, veamos cómo se aplica el atributo **chunksize** en conexiones sql:

```
# Importamos las bibliotecas
import pandas as pd
import teradatasql

# Configuramos los datos de conexión al banco SQL
user = "***"
password = "***"
host = '***'
query="SELECT * FROM FACTURA"

# Abrimos la conexión al Banco
with teradatasql.connect(host=host, user=user, password=password, logmech='LDA
    # Creamos una variable booleana verdadera 'header' para exportar las cabec
    header = True
    # Ejecutamos la query de selección de la tabla 'Factura', pero importamos
    # Para cada 'chunk' (DataFrame con 1MM de filas) exportamos el resultado e
    for chunk in pd.read_sql(query, connect, chunksize=1000000):
        chunk.to_csv('Factura.txt', header=header, index=None, sep='|', mode='
        header = False
```

3. Importando múltiples archivos

Ahora imaginemos que nuestros datos son tráficos de llamadas de celulares, y nos han compartido 2 años de estos datos, el pedido es: “Subir estos 2 años de tráfico a una tabla y dividirla en archivos mensuales, donde cada archivo tendrá el tráfico de un mes”, pero ¿cuál es el problema? Ya no tenemos un único archivo como fuente de origen, sino que tenemos más de 200 archivos y además la suma de todos ellos generará 250MM de líneas, ahora el problema es mayor, veamos cómo encontrarle una solución:

```
# Importando bibliotecas
import pandas as pd
import glob

# Configurando el directorio donde se encuentran nuestros archivos
path = r'D:/files'
all_files = glob.glob(path + "/*")

# Ahora vamos a recorrer uno a uno los archivos de nuestro directorio, y divid
for filename in all_files:
    # Cada archivo será importado al DataFrame 'datos', como no son archivos g
    datos = pd.read_csv(filename, sep = '|', header = None, dtype=str)
    # Crearemos nombres de columnas, porque los archivos no los tienen
    datos.columns = ['OPERATOR_ID','MONTH_DT','CUSTOMER_ID','SUBSCRIBER_ID','M
    # Actualizaremos el valor de la columna 'MONTH_DT' (YYYYMM) con el año-mes
    datos['MONTH_DT'] = datos['TRAFICO_DT'].str[0:6:1]
    # Ahora crearemos un objeto 'grupos' que tendrá una partición(DataFrame) p
    grupos = datos.groupby('MONTH_DT')
    # Ahora que ya tenemos dividido nuestro archivo, vamos a guardar cada part
    for mes,valores in grupos:
        if mes=='201901':
            valores.to_csv(r'D:\resultados\TRAFICO_20190100', header=False, in
        elif mes=='201902':
            valores.to_csv(r'D:\resultados\TRAFICO_20190200', header=False, in
        # Completar para los demás meses del intervalo de 2 años
        elif mes=='202012':
            valores.to_csv(r'D:\resultados\TRAFICO_20201200', header=False, in
        elif mes=='202101':
            valores.to_csv(r'D:\resultados\TRAFICO_20210100', header=False, in
        else:
            valores.to_csv(r'D:\resultados\otros', header=False, index=None, s
```

Conclusiones

Muy bien amigos, ahora ustedes han comprobado como sí es posible trabajar con datos gigantes (Big Data) aprovechando los recursos de la biblioteca Pandas, sea utilizando el atributo **chunksize** o sea dividiendo en **archivos más pequeños**, el objetivo es el mismo, ser capaces de procesar GB de información sin sobrepasar la memoria disponible del ordenador. Por eso, la frase del tema de hoy es: ¡Divide y vencerás! No se olviden de verificar los cursos de Data Science en Alura para aprender más y ¡hasta la próxima!

Puedes leer también:

- Cómo eliminar filas y columnas con Pandas en Python
- Análisis de datos: analizando mi distribución con tres alternativas de visualización
- Matplotlib una biblioteca Python para generar gráficos interesantes

ARTÍCULOS DE TECNOLOGÍA > DATA SCIENCE

- Foro y comunidad exclusiva para resolver tus dudas
- Acceso a todo el contenido de la plataforma por 6 meses

¡QUIERO EMPEZAR A ESTUDIAR!

Paga en moneda local en los siguientes países

ANUAL

US$79,90

un solo pago de US$79,90

- 218 cursos
- Videos y actividades 100% en Español
- Certificado de participación
- Estudia las 24 horas, los 7 días de la semana
- Foro y comunidad exclusiva para resolver tus dudas

Nuevos cursos
cada semana

NAVEGACIÓN

PLANES
INSTRUCTORES
BLOG
POLÍTICA DE PRIVACIDAD
TÉRMINOS DE USO
SOBRE NOSOTROS
PREGUNTAS FRECUENTES

¡CONTÁCTANOS!

¡QUIERO ENTRAR EN CONTACTO!

BLOG

PROGRAMACIÓN
FRONT END
DATA SCIENCE
INNOVACIÓN Y GESTIÓN
DEVOPS

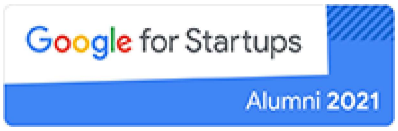
AOVS Sistemas de Informática S.A
CNPJ 05.555.382/0001-33

SÍGUENOS EN NUESTRAS REDES SOCIALES

ALIADOS

Empresa participante do SCALE UP ENDEAVOR

En Alura somos unas de las Scale-Ups seleccionadas por Endeavor, programa de aceleración de las empresas que más crecen en el país.



Fuimos unas de las 7 startups seleccionadas por Google For Startups en participar del programa Growth Academy en 2021

POWERED BY

CURSOS

Cursos de Programación
Lógica de Programación | Java

Cursos de Front End
HTML y CSS | JavaScript | React

Cursos de Data Science
Data Science | Machine Learning | Excel | Base de Datos | Data Visualization | Estadística

Cursos de DevOps
Docker | Linux

Cursos de Innovación y Gestión
Productividad y Calidad de Vida | Transformación Ágil | Marketing Analytics | Liderazgo y Gestión de Equipos | Startups y Emprendimiento